黑马面经

技术点：vue2+router+sass+eslint-stardard+vuex+elementUI+axios

接口文档：<https://www.apifox.cn/apidoc/project-934563/api-19465917>

接口基地址：<http://interview-api-t.itheima.net/>

接口演示：<http://xpzll.gitee.io/hmmj-pc>

Echarts：https://echarts.apache.org/handbook/zh/basics/import/

感触：前端多而杂。

最重要的还是需求分析，分析项目要求。难点还是css呜呜呜---

思路：先弄静态页面再发送请求然后渲染页面。

elementUI内部就是用的scss写的,所以用scss。

这是自己搭架子.vue-cli

vue创建项目：vue create 项目名

运行vue项目脚手架： npm run serve

预处理器：scss/sass(stylus)反人类。css,less

框架+组件库

elementUI网站过时，语法注意下

scss语法$声明变量。---保持多个元素同一个颜色：声明变量 $color:#f00

1. 框架搭建 vue+element 确认技术栈
2. 主题色
3. 封装axios ---设置基地址，请求拦截和响应拦截
4. 本地储存-封装操作token的三个方法：存、取、删

上面是项目准备，一般是项目经理和组长做的搭建项目和大概的架子，没有就才是组员做，

1. 页面组件-位置分析及其页面准备然后进行路由配置（1级和2级也就是登录页和骨架页layout 二级就是数据看板和面经管理）
2. 登录页面完成校验然后封装请求-用户请求页面跳转-导航守卫（判断是否登录token）token失效
3. echarts在Vue中的使用。
4. 渲染面经列表

axios不直接设置基地址 axios.defaults.baseURL=’’

因为后面项目有多个基地址。Import axios from ‘axios’然后const request=axios.create({baseUrl:”,timeout:”]) 然后设置请求拦截和响应拦截。......., 最后暴露出去export default promise

另外：我发请求没反应，如果没有设置超时，大概会等1分钟

存储不习惯用这个，但是后面获取需要这个：const KEY='my-token-element-pc'

存储需要值token传入然后键值对（KEY,value）进行存储，其他不能传值，直接根据键

获取移除。目的：因为在不同页面存储取，所以封装const KEY=这样子方便后期修改

路由配置记得设置路由出口，每个页面最好设置一个名字方便后期排错

Index.vue就代表w这个页面的主页面

路由配置：组件要导入，// 名字-组件-路径，儿子children，{ name: login, path: '/login', component: login }

封装思想：**封装然后暴露出去**

**样式：background:是集合，包含图片位置大小 background:url(‘’) no-repeat center/cover , height:100%vh是高度完全覆盖，然后**

**垂直和水平居中 父级display:flex justiy-content:center align-item:center**

使用 scoped后，父组件的样式将不会渗透到子组件中(组件样式不会成为全局样式造成样式冲突，那一定要加scoped，但是加了scoped样式就作用不到card组件内部浏览器才能看到的内部标签上，所以这时候就需要深度作用选择器)

**深度作用选择器：默认情况下，如果你加了scoped，代表这个样式只能给当前组件写的标签用，也就是某个标签是组件内部的标签（只在浏览器查结构看到的那种）是无法作用到的，要想要样式作用到组件内部，就需要用深度作用选择器**

**语法： ：：v-deep**

**也就是我写的样式和组件里面的潜在样式，加了scoped，组件内部的潜在样式作用不到，就需要在：：v-deep 。作用到组件内部。不专业的叫法叫做样式穿透。Vue官网里面有说法。Scoped-css. 另外一种用法》》》 只能用在原生css里面，不能scss这种非原生预处理器**



垂直居中：line-height:80px ，font-size,color

表单：model这个属性很重要，它跟表单绑定的对象，要想要拿到表单的值，必须先给form加一个model

trigger: 'change' ：页面输入就触发

组件库知识不用背下来，直接看文档学习。框架Vue知识点还是要记忆。

整个表单校验：

原生找dom是documeng,queryselectors(‘选择器’)，首先代码比较长，无法找组件对象。其次它找到的是整个html的对象元素（明明.form是el-form组件，但是它找到的不是组件而是一个dom元素），因为只是想要一部分东西不是全部。



组件库elementUI里面找元素，

找元素然后表单验证

Vue推荐找元素的方式就是ref搭配$refs：要找一个元素就不要用elementUI内置的一些东西，而是用ref，ref起一个名字，用法:标签要找的元素：ref=’别名’ 方法里面：this.$refs.form。里面有组件内置的一些方法和一些属性。加this因为在js里面。ref既可以找dom元素也可以找组件对象，而且找组件内置的一些东西属性方法和数据，写起来比较方便。

要让整个表单校验，首先要找个这个表单对象，然后调用它的validate方法即可做整个表单校验。

elementUI官网没写这个写法：

 {

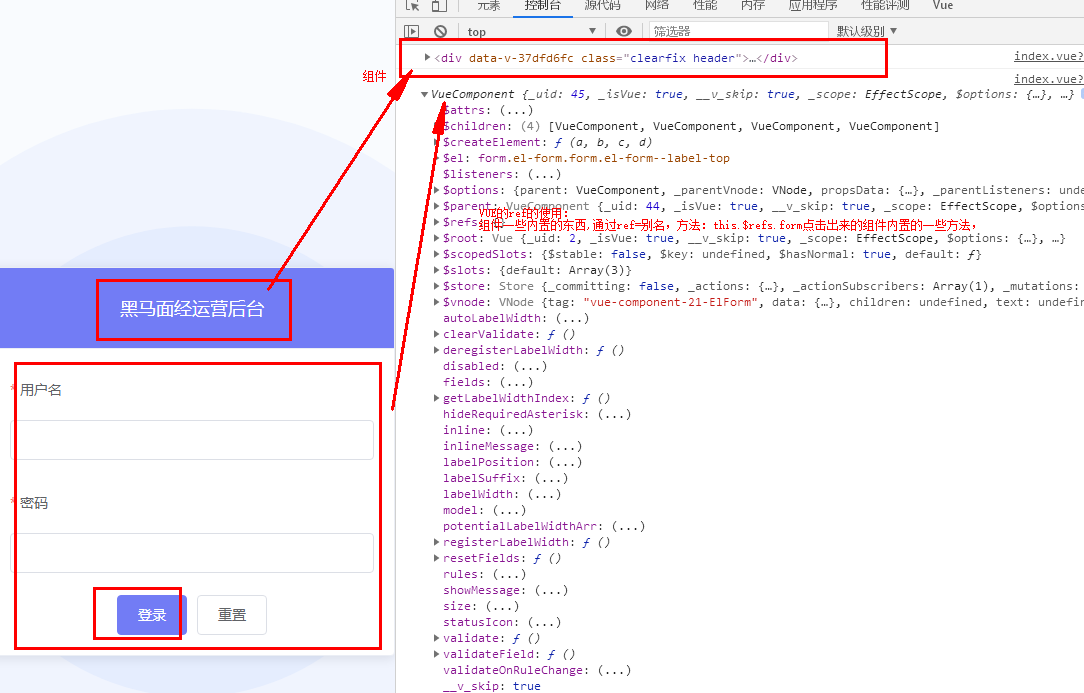
            min: 5,

            max: 11,

            message: '长度在 5 到 11 个字符',

            trigger: ['change', 'blue']

          }





重置就是做两件事：1、把数据恢复到默认值2、清空表单校验结果

封装请求：1、导入基地址2、封装一个用来登录的接口暴露出去在api文件夹里面封装user.js

账号和密码:admin8 密码admin

本地储存token：

vuex的特点：数据是响应式的,（数据改变页面也跟着改变），vue里面直接使用Vuex，本地储存要再建个变量才行，缺点就是一刷新就恢复到默认值,vuex恢复到默认值，本地储存就按储存的显示

本地储存缺点：数据不是响应式的，，在vuex的组件里无法直接使用。改变本地储存页面没有发生改变。优点：数据长久保存，只要自己不删除就一直都在。

本地储存+Vuex组合使用：1、如果一个项目一个数据多个组件使用，最好存到vuex里面，2、另外这个数据下次打开或者刷新之后还要在，那就利用本地储存+vuex

   <p>{{ $store.state.count }}</p>

    <p>{{ count }}</p>

应用：token要长期保存，另外需要在其他组件长期使用。根据之前学的模块化思想，Vuex的数据最好不要放到根store里面，最好放到模块里面命名

vuex忘记了模块化的使用



本地存储到store里面的user.js

 token: getToken() || ''

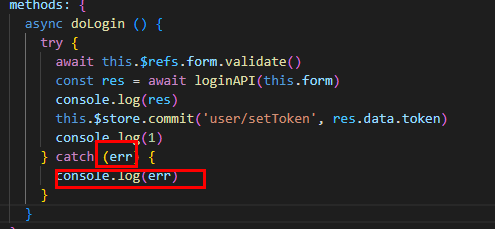
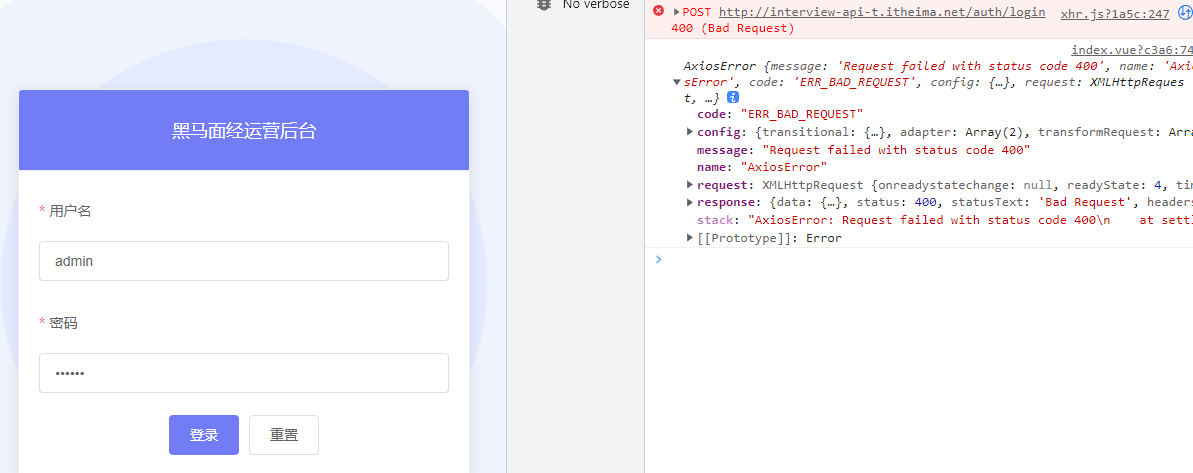
一刷新vuex就没有了，所以就需要在state里面直接获取本地存储变为默认值或者空值未登录。

Try.catch捕获异常。

Catch{}要单独判断到底是哪个错，如果是请求错才弹出提示

发送请求trycacth捕获异常axios错误这缺失弱点。

加catch弹出错误提示，最好在响应拦截里的错误拦截需要做一个统一的处理。

 console.dir(error)

这个代表什么含义。

请求失败统一弹出提示-位置注意：其他错都弹出，不能准确判断是否是请求出错还是其他异常，所以在请求拦截器里面做处理从而确定是请求问题。

原因：登录账号或密码错误会引发服务器响应报错。而我们现在没有任何的提示告诉用户到底怎么了，所以要弹出提示。但是如果每次发请求都加catch再弹出提示比较麻烦，而且有个问题try包的代码任何一句出错都可能会进到catch里，那在catch里还得单独判断到底是不是请求报错，再弹出提示，麻烦

处理方式：基于上面的问题，所以如果我们要弹出提示，最好写到响应拦截的错误拦截里，这样就只要写一次代码，以后任何请求报错都能弹出提示

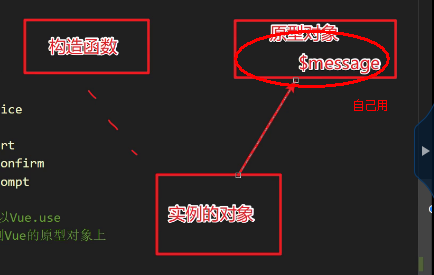
this.message.success是在Vue实例也就是在vue文件里面，在js文件也就不在Vue实例里面，需要导入import {Message}from ‘element-ui’然后Massage.error(err.message.data.)这样子才能使用。

有个疑问：在谷歌浏览器运行和在ie浏览器运行效果不一样，有时候一个有弹框一个没有弹框？

elmentui的按需导入，如果是组件就可以Vue.use。如果是方法就需要手动做挂载，把它挂载到Vue的原型对象上，这样子所有Vue实例才能直接使用。要区分是组件还是方法。这种按需导入需要考虑的情况。全局导入就不需要考虑这种情况。



导航守卫-判断（门卫）：我没有登录也就没有token也可以，这时候就需要导航守卫进行判断了。后台的登录要做权限判断。白名单。只要是登录（有token）就放行，其他情况打回登录页面



全局前置守卫-router 满足条件：登录页或者其他有token，不满足：只能去登录页

router.beforeEach((to, from, next) => {

  if (to.path === '/login' || getToken()) {

    next()

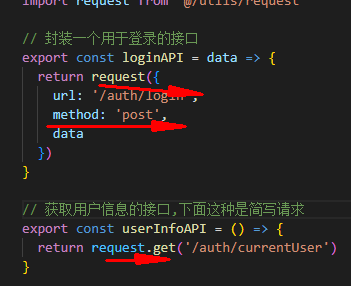
  } else {

    next('/login')

    Message.warning('请先登录')

  }

})



发送请求拦截：

请求携带token:明明有token却报没有找到正确的token，这时候就看接口文档排查。已经存储到本地存储和vuex，但是发请求没有携带token，发送请求都要携带token，发一次请求都要加一次token，简便方法就是请求拦截器：把请求拦截下来然后给它加token。加之前先判断是否有token再加。

请求拦截器+请求头：config.headers.Authorization = 'Bearer' +token

主要看接口格式要求。一般接口加了Beare，大概率是后面有空格，主要还是看后端接口文档决定。



解构语法：const obj={name=”jack”}

报401错误也就是没有携带token或者请求错误。后面请求都要携带token，所以添加在请求拦截器里面。

退出：elementUI组件退出不是真正的退出，不仅仅只是跳转到login页面，同时还需要删除本地token和vuex里面的token。

vuex数据删除:通过删除mutations里面的操作修改,加一个方法,logout(state){state.token=“”}然后导入removeToken方法。removeToken(),最后在页面vue使 用 ,this.$store.commit(‘user/logout’)。

Token过期-被动处理：也就是401状态。业务：token都是有有效期的，需要携带token的请求会报401错误，会在响应拦截里面体现进行判断。过期就应该弹出登录转态时效请重新登录然后打回登录页面。不能用this.$router.push()因为是给Vue实例用的，所以需要导入然后使用。然后再删除本地和Vuex里面的token，就直接调用store.user.js里面的logout()方法直接删除。

import { router } from '@/router/index'

 router.push('/')

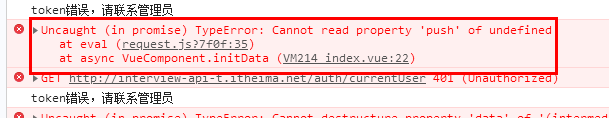
this.$store.commit('user/logout')是给Vue实例对象用的，所以在request.js不能直接使用。

解决办法:1、导入import store from '@/store'

2、使用：store.commit('user/logout')

import { router } from '@/router'和import router from '@/router'的区别

导致出现push提示错误



页面不停的刷新，vscode没有操作，浏览器不停的刷新页面，啥原因？

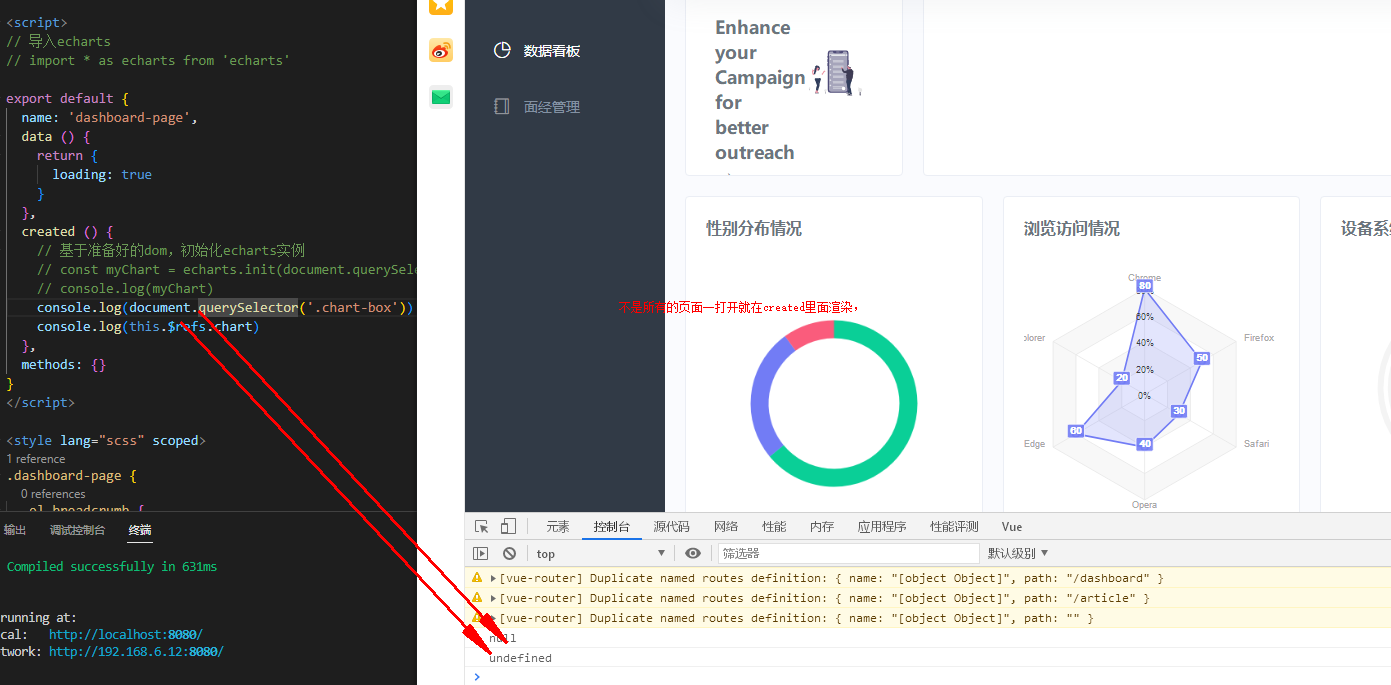
什么时候用console.dir什么时候使用console.log()

Echarts在vue中的使用：

1、第一步先去官网看快速上手使用手册，下载导入按需使用。

2、在Vue中不建议使用document.queryselector而是使用ref。另外这个图标要一打开就显示，所以就需要写在created。但是createdm没用，所以需要用mounted。最早访问dom的钩子。知识点：页面渲染的时机。echart官网的document.queryselector可以用但是不建议用Vue。

3、实例化echarts对象。最后就是复制案例，修改标题然后看echart里面的api的配置项手册。嵌套对象用法。不知道哪个是哪个就注释掉。有几个echarts就有几个配置和对象。一个对象对应一个配置，不能一对多。







**Element -面包屑和按钮的使用**



表格和表单的区分。

另外用了elementui就尽量在里面找样式，不要自己单独写，



elementUI表单渲染：

:data="tableData"这个绑定的tableData必须是数组，空数组显示暂无数据。如果绑定的是对象就会报错。

封装面经列表-发送请求获取数据（article.js）：按需导入方法。导入基地址然后暴露封装接口请求，在Vue页面导入方法然后调用方法（数据），对数据进行处理，为空数组赋值

方法（函数）一定记得加括号！！！

有括号是方法，没括号是变量！！！ 导入不同的形式

import { articleListAPI } from '@/api/article'

然后希望什么时候有数据-页面一打开就有数据。所以写在created。前面加个async,await

命名要小驼峰语义化

什么时候用return,什么时候不用return。

函数和方法的区别：函数和方法本质上是一样的，只不过方法是函数的特例，是将函数值赋给了对象。

https://blog.csdn.net/qq\_44163269/article/details/104627274?ops\_request\_misc=%257B%2522request%255Fid%2522%253A%2522167541750516800182798644%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request\_id=167541750516800182798644&biz\_id=0&utm\_medium=distribute.pc\_search\_result.none-task-blog-2~all~sobaiduend~default-1-104627274-null-null.142^v73^insert\_down4,201^v4^add\_ask,239^v1^insert\_chatgpt&utm\_term=%E5%89%8D%E7%AB%AF%E6%96%B9%E6%B3%95%E5%92%8C%E5%87%BD%E6%95%B0%E5%8C%BA%E5%88%AB&spm=1018.2226.3001.4187

图标使用：用button找不到对应样式，那就用elementui的a-link标签。然后图标偏小，饿了么官网没有这设置，所以就自己写。设置样式大小跟文字处理一样：style=’font-size:18px’。行内式。

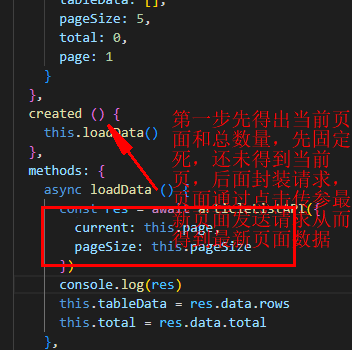
elementUi提供分页使用：

效果不显示很有可能是填错位置。

大概就是知道页面效果用的elementui哪个组件（比如layout就是设置分页的布局，prev代表上一页，pager代表页码，next代表下一页。total设置总条数）用组件的同时不要忘记原生写法不然换框架就玩蛋了。



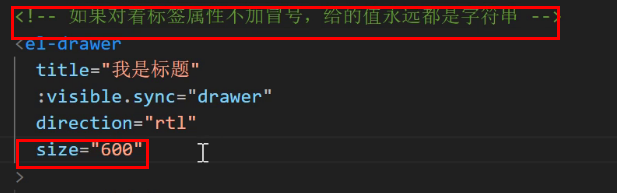
分页功能：点击改变页码，就需要重新发送请求：根据每页页码获取当前页数据。接口是完全一样的，只是请求参数页码不一样，所以对created里面的页面就进行封装，封装成一个方法方便别的地方进行调用。然后在created调用即可。然后再完成页码的点击时间。



路由重定向：默认跳转

  { path: '/', redirect: '/dashboard' },当我输入/直接跳转到dashboard

如果对着标签属性不加冒号，给的值永远是字符串，传递的数据都是字符串



抽屉分析：点击-预览和修改，点击之后都弹出抽屉，可以触发同一个方法，但是细节又不一样，所以还得传入不同的参数区分。

标题不能写死，前期是写死的，先设置默认值，然后方法传参赋值进行保存，最后用计算属性对数据进行判断处理

。

对象里面的数据结合接口文档参数名字一致性。

富文本编辑器的使用：elementui没有富文本插件，跟Vue契合的有

[https://www.npmjs.com/package/vue-quill-editor](https://www.npmjs.com/package/vue-quill-editor" \t "https://www.wolai.com/_blank)，使用看链接官网地址说明。

Ssr:1、Server-Side Rendering我们称其为SSR,意为服务端渲染

用的是spa。

默认情况下规则只能用在表单元素上，如果用在富文本上要单独写代码进行校验。根据文档查询可得，表单对象有个方法叫validateFields。因为我们需要失去焦点就校验，所以给富文本加@blur=’$refs.form.validateFields‘’

富文本编辑器：有个小问题，没有非空校验。原因：校验规则是给表单元素进行校验的，但此时富文本不是表单元素，所以element应该也考虑到了不会用到原生组件库情况。所以去elemetui表单元素方法位置查看-对部分表单元素进行校验。手动校验。单独校验：[在标签上写上@blur=”$refs.form.validateFields（‘content’）”](mailto:在标签上写上@blur=\”$refs.form.validateFields（‘content’）\”) 当失去焦点叫调用 表单的校验方法。

validate校验，try catch的使用，async await的使用。

新增功能：

校验表单：点击，@click=”save” 方法：

   async save () {

      try {

        //整个表单校验

        await this.$refs.form.validate()

      // 能来到这，证明校验通过，就发起请求做新增。api里面的article.js

      } catch {}

    }



传参this.form，然后接收这个返回值const res=

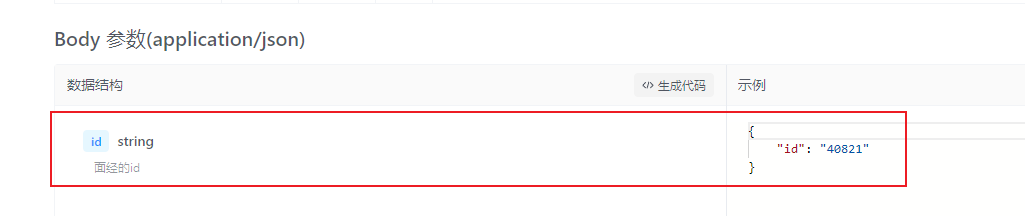
后面出现新增成功提示，然后关闭弹窗，最后重新渲染数据再加载一次。之前封装了一个lodaDate()这个直接重新调用一次。thisllodaDate()。这就是封装方法的好处了。新增渲染之后页码要跳转到第一页，所以this.page=1

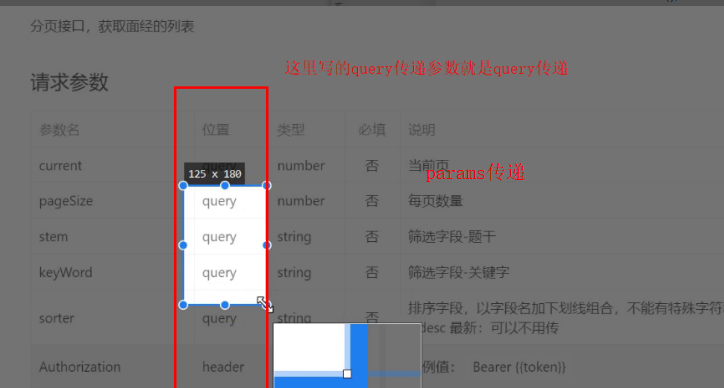
存在问题：关闭重新点击新增-内容还在，下次打开抽屉，上一次内容还在

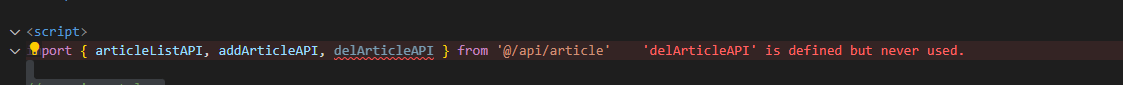
解决：每次关闭都把内容清空（也就是把表单重置。但是会有Bug）

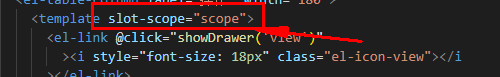
步骤：需要在抽屉关闭前，先清空内容再关闭，意味着要给抽屉加一个关闭前的回调。（done）后面调用done(),

Body参数用data传递？？

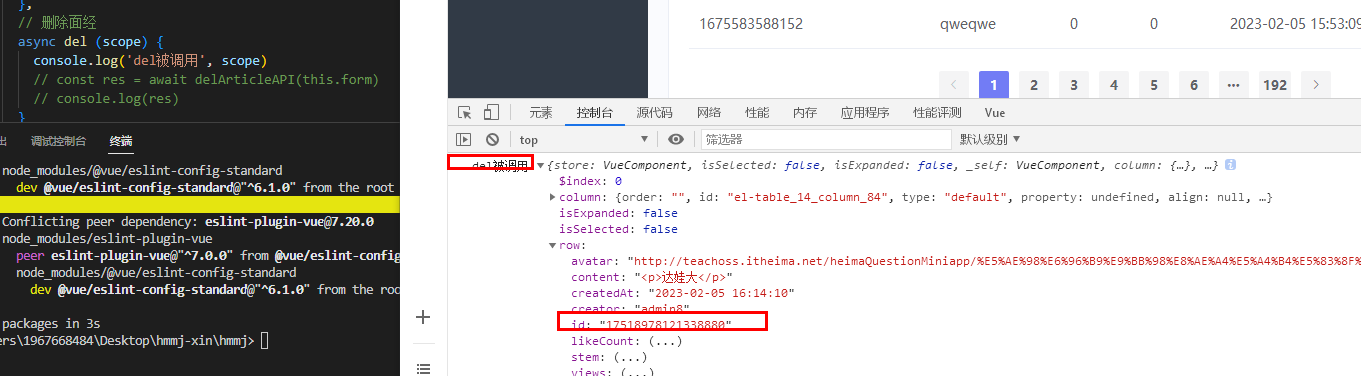








这就是伏笔，先打印看看是什么东西，找到id然后删除



scope有当前这一行的相关信息，其中scope.row就是这一行的数据

删除bug:el-table之前用了自定义列，发现里面有个template slot-scope=‘’经过代码观察发现scope就是代表当前这行的相关信息，其中有个很重要的属性就叫做scope.row，就有这行数据.传参scope.row.id，方法（id）

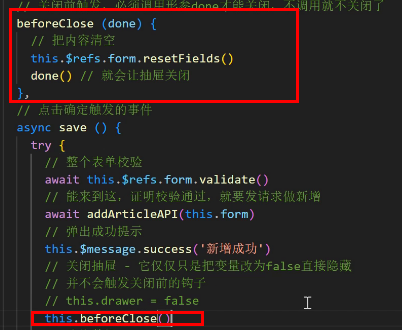
最后一页如果只剩一条了，你再删除，就会显示暂无数据。最后一条删除，意味着这一页不存在，请求还是最后一页。

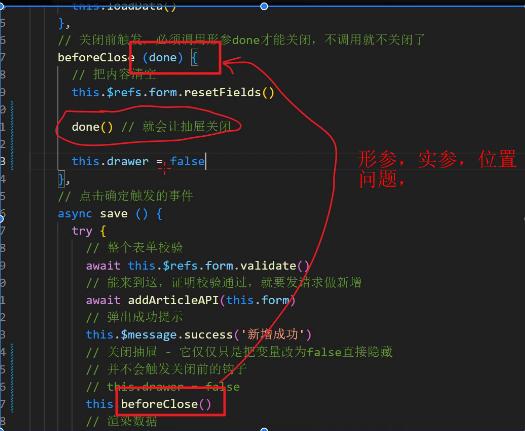
删除有一个bug，当前5列，5列删除完之后最后一列就变成暂无数据，。因为当前页删除完之后，后面还是发送这一页的请求，所以是暂无数据。

解决：当前数组长度为1.就，也就是this.tabeData.length==1,this.page--页码减减。并且这一页不等于第一页。

this.tableData.length === 1 && this.page !== 1

点击新增-确定按钮，内容没有清空，没有触发beforeClose()只是把变量改为false，改为方法调用顺便清空了。





新增成功不清空内容的Bug:新增成功并不会清空内容，原因：直接把drawer改为false,不调用beforeClose这个回调函数。解决：要关闭就自己手动调用beforeClose,并且在beforeClose里面加this.drawer=false即可。

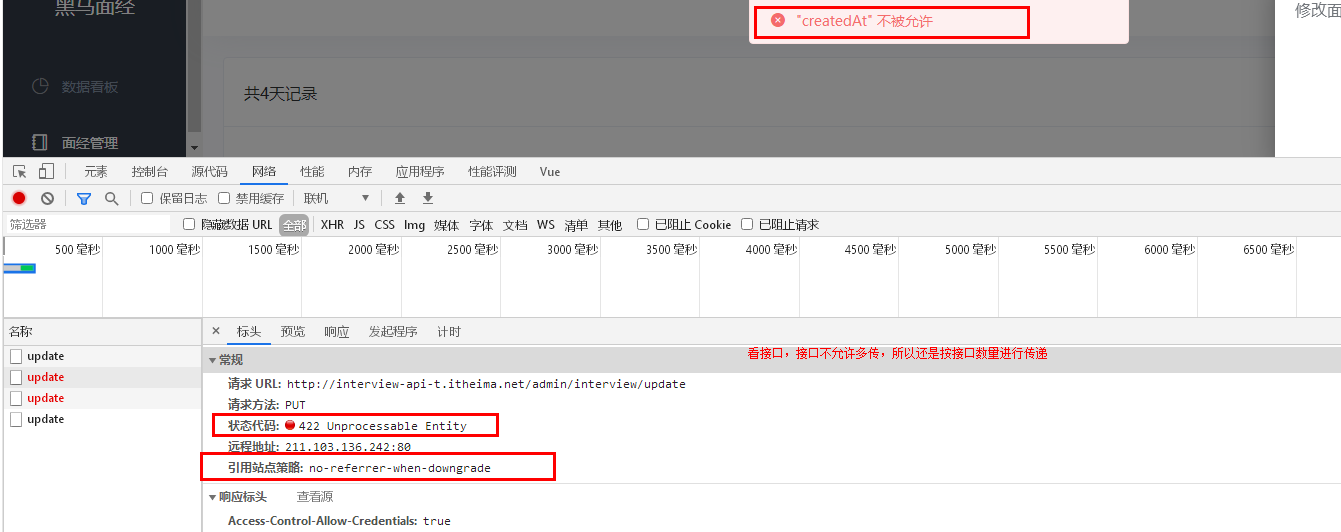
修改功能：根据id查出这篇详情，然后再渲染到页面上。把请求到的数据给表单双向绑定的form。form一变页面也跟着改变。

restful风格接口风格？

知道什么时候需要接口，什么时候需要封装。BOdu参数需要区分data传递还是params传递。

修改-发请求就需要封装一个接口

新增和修改的区别是：1、一个有id,一个没有id。2、另外一个是标题也不一样。3、type也不一样。所以save加个判断，要么是新增要么是修改，预览只是给你看的。





修改功能：修改意味着要加点击时间，但确定按钮早就加了点击事件，只不过之前是做新增，现在意味着既要做新增又要做修改，所以就做一个判断。所以对部分代码做判断。只有新增跳到第一页，其他操作当前页面操作就在当前页面。

另外还有一个bug（去elementui找线索）:点击修改完成之后，再点击新增，页面还是显示的修改页面。

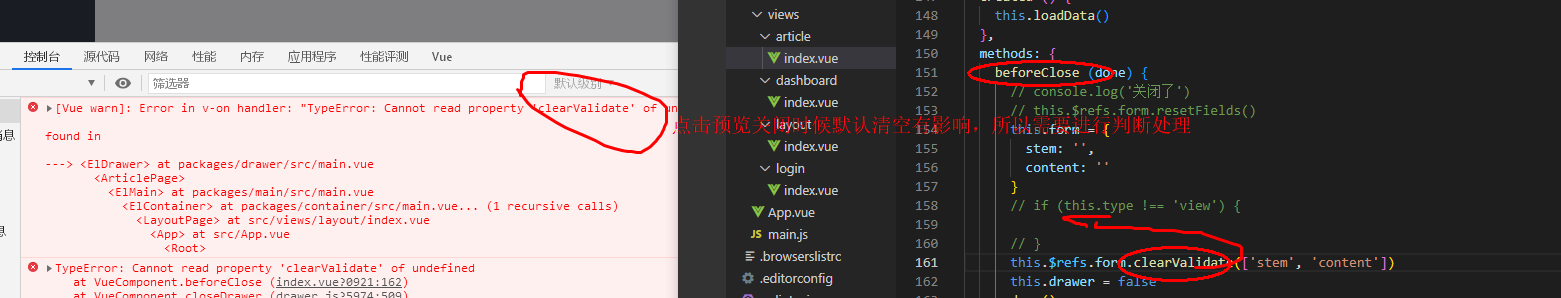
原因：resetFields只是把内容重置成默认，并没有清空。点击编辑是有默认值Vuex。

那就把.this.$refs.form.resetFields()注释掉改为this.form={stem:”,content:”}这也不行，因为校验规则没有清空，只是把值清空了。所以还得需要这个方法this.$refs.form.clearValidate([‘stem’,’content’])()（因为既可以重置内容又可以清空校验规则）

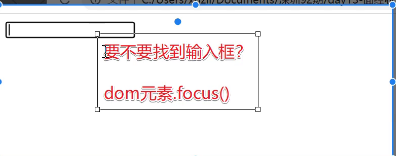


预览功能-v-if v-else

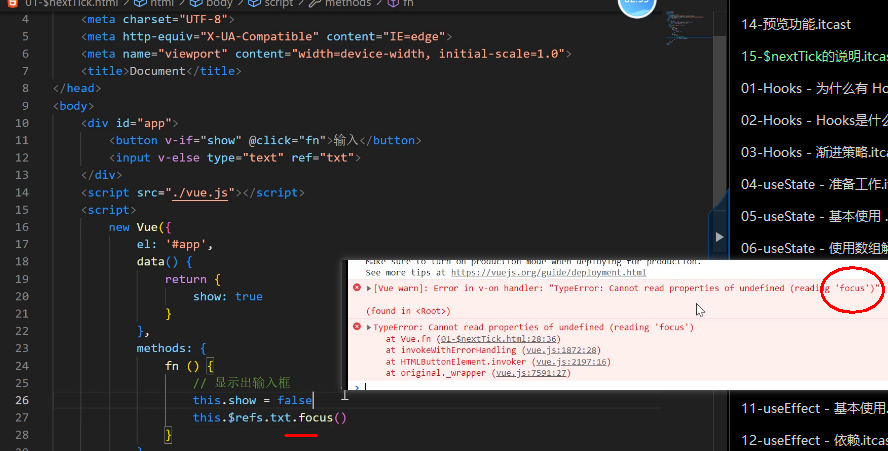
在抽屉里准备一个div，而这个div默认不用显示，只有当做预览才显示，其他情况都显示表单，但是此时发消息点击空白关闭时会报错，原因是此时没有渲染出表单，而关闭的方法会调用表单的clear方法，所以要做判断



$nextTickh说明



在vue中加$ref。





并没有获得焦点，然后直接打印console.log(this.$refs.txt)结果是undefined。

V-if是懒加载，显示时候才会渲染，没显示时都不会渲染。所以页面一打开只渲染button，没有渲染input，照理说这句话以后界面就会有input。

原因：Vue中的dom渲染是异步的,，只有同步弄完才搞异步渲染，所以setTimeout（）

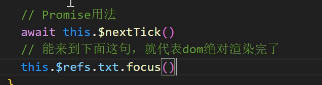
可以实现，但是不建议，是因为可能前面也有异步队列操作，时间不明确，另外阅读没那么好，所以Vue建议，你需要等dom渲染完后就调用，那就用updated（也不行,但凡数据发生改变，而且dom渲染完就调用。，但是如果你把一些耗费性能的东西写在updated里面-它有性能问题），所以Vue提供另外一个选择$nextTick，它会传入一个回调函数

语法： this.$nextTick(()={ this.$refs.txt.focus()})只有你点击了这次才会调用。



Promise用法

This.$nextTick() 能来到下面这句就代表dom绝对渲染完成了。



node和es6学得不好，很多都忘记了